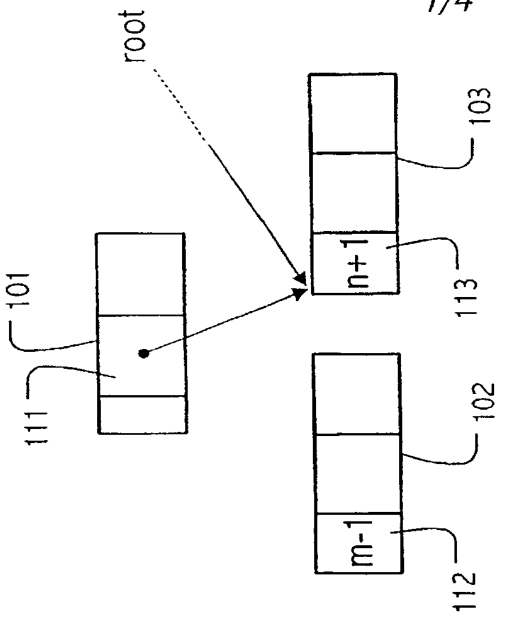
root
101
111
103
n+1
113
102
m-1
112

FIG. 1B

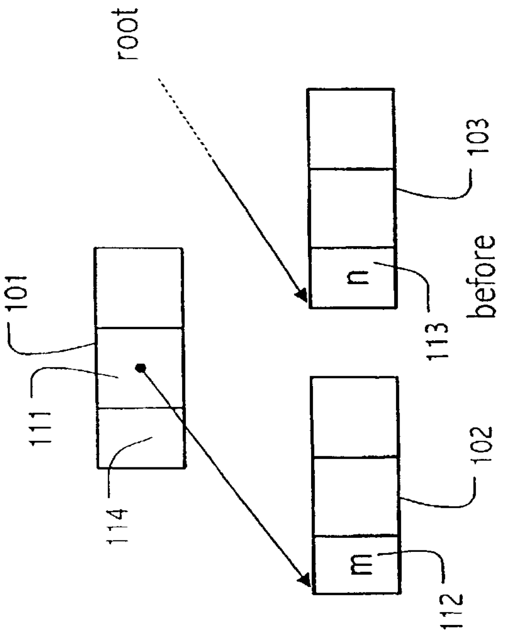
root
101
111
114
103
n
113
before
102
m
112

FIG. 1A

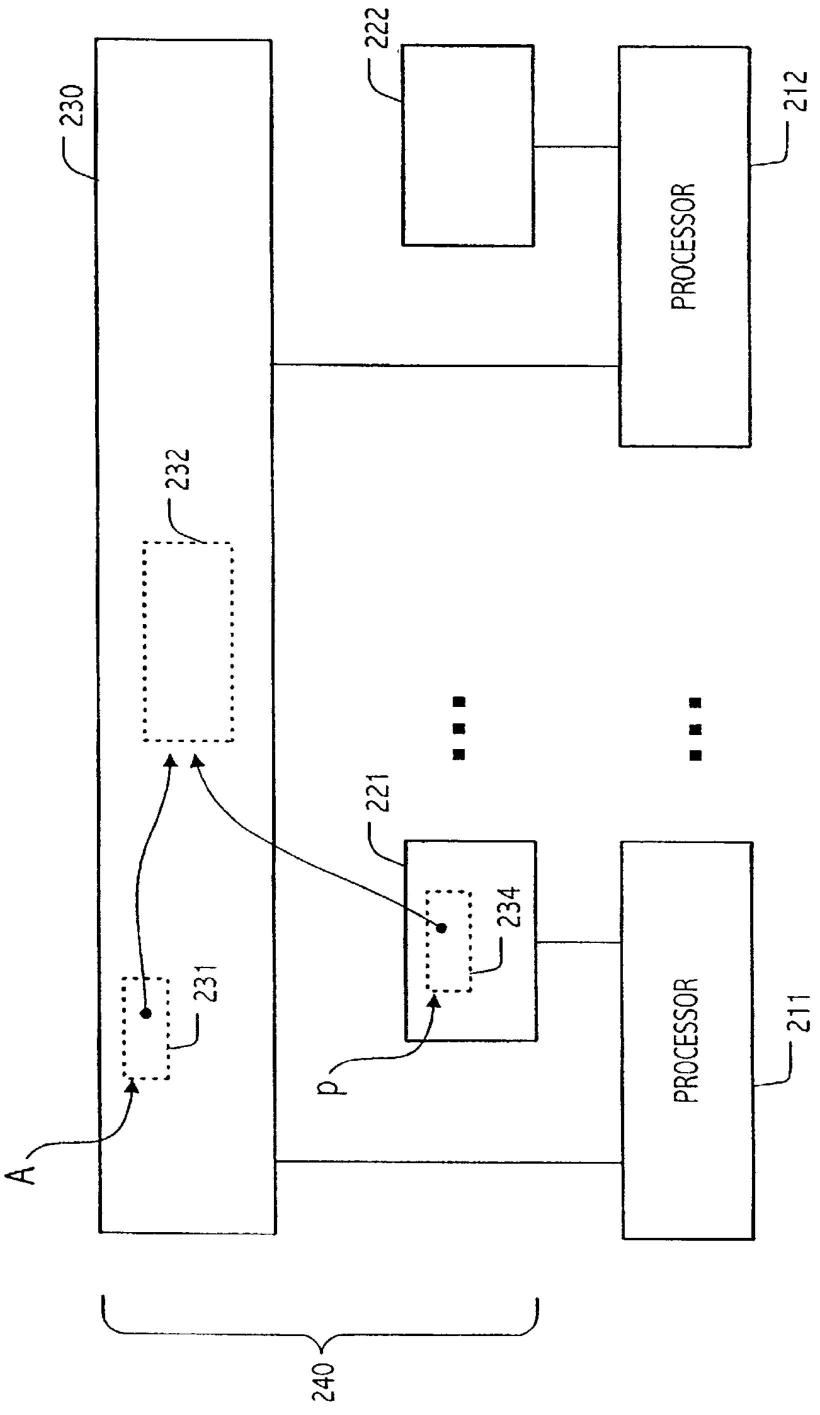

FIG. 2

```
class SNode {
1   class SNode *L, *R;  valtype V;
2   SNode() {};
};

class Snark {
3   SNode *Dummy, *LeftHat, *RightHat;
4   Snark() {

5      Dummy = new SNode;
6      Dummy→L = Dummy;
7      Dummy→R = Dummy;
8      LeftHat = Dummy;
9      RightHat = Dummy;
    };

10  valtype pushRight(valtype v);
11  valtype pushLeft(valtype v);
12  valtype popRight();
13  valtype popLeft();
};
```

FIG. 3A

```
class SNode {
31  class SNode *L, *R;  valtype V;  long rc;
32  SNode() : L(Null), R(Null), rc(1) {};
};

class Snark {
33  SNode *Dummy, *LeftHat, *RightHat;
34  Snark() : Dummy(Null),
                      LeftHat(Null), RightHat(Null) {
35     LFRCStoreAlloc(&Dummy, new SNode);
36     LFRCStore(&Dummy→L, Null);
37     LFRCStore(&Dummy→R, Null);
38     LFRCStore(&LeftHat, Dummy);
39     LFRCStore(&RightHat, Dummy);
    };
40  ~Snark() {
41     while(popLeft()!=EMPTYval);
42     LFRCStore(&Dummy, Null);
43     LFRCStore(&LeftHat, Null);
44     LFRCStore(&RightHat, Null);
    };
45  valtype pushRight(valtype v);
46  valtype pushLeft(valtype v);
47  valtype popRight();
48  valtype popLeft();
};
```

FIG. 4A

```
valtype Snark::pushRight(valtype v) {
 SNode *nd = new SNode;
 SNode *rh, *rhR, *lh;
 if (nd == Null)

   return FULLval;

 nd→R = Dummy;
 nd→V = v;
 while (true) {
   rh = RightHat;
   rhR = rh→R;
   if (rhR == rh) {
     nd→L = Dummy;
     lh = LeftHat;
     if (DCAS(&RightHat, &LeftHat, rh, lh, nd, nd))


       return OKval;

   } else {
     nd→L = rh;
     if (DCAS(&RightHat, &rh→R, rh, rhR, nd, nd))


       return OKval;
}  }  }
```

FIG. 3B

```
valtype Snark::pushRight(valtype v) {
 SNode *nd = new SNode;
 SNode *rh = Null, *rhR = Null, *lh = Null;
 if (nd == Null) {
   LFRCDestroy(rhR, nd, rh, lh);
   return FULLval;
 }
 LFRCStore(&nd→R, Dummy);
 nd→V = v;
 while (true) {
   LFRCLoad(&RightHat, &rh);
   LFRCLoad(&rh→R, &rhR);
   if (rhR == Null) {
     LFRCStore(&nd→L, Dummy);
     LFRCLoad(&LeftHat, &lh);
     if (LFRCDCAS(&RightHat, &LeftHat,
                               rh, lh, nd, nd)) {
       LFRCDestroy(rhR, nd, rh, lh);
       return OKval;
     }
   } else {
     LFRCStore(&nd→L, rh);
     if (LFRCDCAS(&RightHat, &rh→R,
                              rh, rhR, nd, nd)) {
       LFRCDestroy(rhR, nd, rh, lh);
       return OKval;
}  }  }  }
```

FIG. 4B